AFRL-AFOSR-UK-TR-2013-0042

Enhancing the Dependability of Complex Missions Through Automated Analysis

David S. Rosenblum
Fokion Zervoudakis

University College London UCL
Gower Street
London WC1E6BT UNITED KINGDOM

EOARD Grant 10-3007

Report Date: September 2013

Final Report from 15 January 2010 to 14 July 2013

Distribution Statement A: Approved for public release distribution is unlimited.

Air Force Research Laboratory
Air Force Office of Scientific Research
European Office of Aerospace Research and Development
Unit 4515 Box 14, APO AE 09421

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.
PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
03 September 2013	Final Report	15 January 2010 – 14 July 2013

4. TITLE AND SUBTITLE

Enhancing the Dependability of Complex Missions Through Automated Analysis

5a. CONTRACT NUMBER

FA8655-10-1-3007

5b. GRANT NUMBER

Grant 10-3007

5c. PROGRAM ELEMENT NUMBER

61102F

6. AUTHOR(S)

David S. Rosenblum
Fokion Zervoudakis

5d. PROJECT NUMBER

5d. TASK NUMBER

5e. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University College London UCL
Gower Street
London WC1E6BT UNITED KINGDOM

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

EOARD
Unit 4515
APO AE 09421-4515

10. SPONSOR/MONITOR'S ACRONYM(S)

AFRL/AFOSR/IOE (EOARD)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

AFRL-AFOSR-UK-TR-2013-0042

12. DISTRIBUTION/AVAILABILITY STATEMENT

Distribution A: Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The increasing complexity of current uninhabited aerial vehicle (UAV) missions is overwhelming human mission developers, and automated mission planning systems and simulation environments. The objective of this project was to enhance the dependability of complex UAV missions with the application of modern automated program analysis techniques. The key insight supporting this project was the treatment of a mission plan as a type of software or software representation, which could be analyzed with software verification methods, including model checking and probabilistic model checking, to detect potential errors before mission execution. The key contribution of this work is a novel method for domain-specific model checking called "cascading verification." The method uses composite reasoning over high-level system specifications and formalized domain knowledge to synthesize both low-level system models and the behavioral properties that need to be verified with respect to those models. In particular, model builders use a high-level domain-specific language (DSL) to encode system specifications that can be analyzed with model checking. Domain knowledge is encoded in the Web Ontology Language (OWL), the Semantic Web Rule Language (SWRL) and Prolog, which are combined to overcome their individual limitations. Synthesized models and properties are analyzed with the probabilistic model checker PRISM. Cascading verification is illustrated with a prototype system that verifies the correctness of UAV mission plans. An evaluation of this prototype reveals non-trivial reductions in the size and complexity of input system specifications compared to the artifacts synthesized for PRISM.

15. SUBJECT TERMS

EOARD, mission planning, software engineering, computational modeling

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18, NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Gregg Abate
UNCLAS	UNCLAS	UNCLAS	SAR	15	19b. TELEPHONE NUMBER *(Include area code)* +44 (0)1895 616021

European Office of Aerospace Research & Development

Grant FA8655-10-1-3007

Enhancing the Dependability of Complex Missions Through Automated Analysis

Program Manager: James Lawton, Ph.D.

Final Report

PI: Prof. David S. Rosenblum

School of Computing
National University of Singapore

Ph.D. Student: Fokion Zervoudakis

Department of Computer Science
University College London

August 2013

Distribution A: Approved for public release; distribution is unlimited.

Contents

Distribution A: Approved for public release; distribution is unlimited.

1 Summary

The increasing complexity of current *uninhabited aerial vehicle* (UAV) missions is overwhelming human mission developers, and automated mission planning systems and simulation environments. The objective of the project we carried out under our EOARD grant was to enhance the dependability of complex UAV missions with the application of modern automated program analysis techniques. The key insight supporting this project was the treatment of a mission plan as a type of software or software representation, which could be analyzed with software verification methods—including model checking and probabilistic model checking—to detect potential errors before mission execution.

Model checking is an established formal method for verifying the desired behavioral properties of system models. But popular model checkers tend to support low-level modeling languages that require intricate models to represent even the simplest systems. Modeling complexity arises in part from the need to encode *domain knowledge*—including domain objects and concepts, and their relationships—at relatively low levels of abstraction. Our research demonstrates that, once formalized, domain knowledge can be reused to enhance the abstraction level of model and property specifications, and the effectiveness of probabilistic model checking.

A refereed conference paper, which was accepted for publication by the 9th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2013), describes a novel method for domain-specific model checking called *cascading verification*. The method uses composite reasoning over high-level system specifications and formalized domain knowledge to synthesize *both* low-level system models and the behavioral properties that need to be verified with respect to those models. In particular, model builders use a high-level *domain-specific language* (DSL) to encode system specifications that can be analyzed with model checking. Domain knowledge is encoded in the *Web Ontology Language* (OWL), the Semantic Web Rule Language (SWRL) and Prolog, which are combined to overcome their individual limitations. Synthesized models and properties are analyzed with the probabilistic model checker PRISM. Cascading verification is illustrated with a prototype system that verifies the correctness of UAV mission plans. An evaluation of this prototype reveals non-trivial reductions in the size and complexity of input system specifications compared to the artifacts synthesized for PRISM.

The remainder of this report describes the work accomplished in the three years of the project—including cascading verification and its application to the analysis of complex UAV missions. Research was funded by the Air Force Office of Scientific Research and the European Office of Aerospace Research & Development. The sudden arrival of the grant in January 2010 and the difficulty in recruiting an able Ph.D. student for the grant (Fokion Zervoudakis) meant that the work described here began in earnest in July 2010. This led us to request a six-month no-cost extension to the grant, which was approved. The project involved collaboration between Fokion Zervoudakis at University College London (UCL), David S. Rosenblum at the National University of Singapore (NUS), and Sebastian Elbaum at the University of Nebraska-Lincoln

Distribution A: Approved for public release; distribution is unlimited.

(UNL), who was funded separately under a grant from AFOSR.

2 Introduction

Model checking is an established formal verification method whereby a model checker systematically explores the state space of a system model to verify that each state satisfies a set of desired behavioral properties [1].

Research in model checking has focused on enhancing the efficiency and scalability of verification by employing partial order reduction, and by exploiting symmetries and other state space properties. This research is important because it mitigates the complexity of model checking algorithms, thereby enabling model builders to verify larger, more elaborate models. But the complexity associated with model and property specification has yet to be sufficiently addressed. Popular model checkers tend to support low-level modeling languages that require intricate models to represent even the simplest systems. For example, PROMELA—a language used by the model checker Spin—is essentially a dialect of the relatively low-level programming language C. Another example is the modeling language used by the probabilistic model checker PRISM. Due to lack of appropriate control structures, the PRISM language forces model builders to pollute model components with variables that act as counters. These variables are manipulated at runtime to achieve desirable control flow from otherwise unordered commands.

Modeling complexity arises in part from the need to encode *domain knowledge*—including domain objects and concepts, and their relationships—at relatively low levels of abstraction. We will demonstrate that, once formalized, domain knowledge can be reused to enhance the abstraction level of model and property specifications, and the effectiveness of probabilistic model checking.

Leveraged appropriately, formal domain knowledge can decrease specification and verification costs. On the verification side, the model checking framework Bogor achieves significant state space reductions in model checking of program code by exploiting characteristics of the program code's deployment platform [2]. On the specification side, *semantic model checking* supplements model checking with semantic reasoning over domain knowledge encoded in the *Web Ontology Language* (OWL). Semantic model checking has been used to verify Web services [3, 4]; Web service security requirements [5]; probabilistic Web services [6]; Web service interaction protocols [7]; and Web service flow [8]. Additionally, multi-agent model checking has been used to verify OWL-S process models [9].

OWL is a powerful knowledge representation formalism, but expressive and reasoning limitations constrain its utility in the context of semantic model checking; for example, OWL cannot reason about triangular or self-referential relationships. The Semantic Web Rule Language (SWRL)—a W3C-approved OWL extension—addresses some of these limitations by integrating OWL with Horn-like rules. But, like OWL, SWRL cannot reason effectively with negation. The *logic programming* (LP) language Prolog can be used to overcome problems that are intractable

Distribution A: Approved for public release; distribution is unlimited.

in OWL+SWRL. Prolog, however, lacks several of the expressive features afforded by OWL, including support for equivalence and disjointness.

We have designed a novel method for domain-specific model checking called *cascading verification* [10]. Our method uses composite reasoning over high-level system specifications and formalized domain knowledge to synthesize *both* low-level system models and the behavioral properties that need to be verified with respect to those models. In particular, model builders use a high-level *domain-specific language* (DSL) to encode system specifications that can be analyzed with model checking. A *compiler* uses automated reasoning to verify the consistency of each specification with respect to domain knowledge encoded in OWL+SWRL and Prolog, which are combined to overcome their individual limitations. If consistency is deduced, then explicit and inferred domain knowledge is used by the compiler to synthesize both a *discrete-time Markov chain* (DTMC) model and *probabilistic computation tree logic* (PCTL) properties from template code. PRISM subsequently verifies the model against the properties. Thus, verification *cascades* through several stages of reasoning and analysis.

Our method gains significant functionality from each of its constituent technologies. OWL supports expressive knowledge representation and efficient reasoning; SWRL extends OWL with Horn-like rules that can model complex relational structures and self-referential relationships; Prolog extends OWL+SWRL with the ability to reason effectively with negation; DTMC introduces the ability to formalize probabilistic behavior; and PCTL supports the elegant expression of probabilistic properties.

Cascading verification is illustrated with a prototype system that verifies the correctness of *uninhabited aerial vehicle* (UAV) mission plans. We used this prototype to analyze 58 mission plans, which were based on real-world mission scenarios developed independently by DARPA [11] and the Defense Research and Development Canada (DRDC) agency [12]. As an implementation of cascading verification, our prototype realized a non-trivial reduction in the effort required to specify system models and behavioral properties. For example, from 23 lines of YAML code comprising 92 tokens, cascading verification synthesized 104 lines of PRISM code comprising 744 tokens and three behavioral properties (with our prototype, model builders encode mission specifications in a domain-specific dialect of the human-readable YAML format [13]).

3 Methods, Assumptions and Procedures

The problem outlined in the previous section cannot be addressed with a single technology. Our solution was to develop a method that integrates OWL+SWRL, Prolog and DTMC and PCTL. OWL was chosen because it is an established knowledge representation formalism, and the ontology specification language recommended by the W3C [14]. OWL limitations motivate several contending extensions, including SWRL, CARIN, $\mathcal{AL}$-log, DL-safe rules, $\mathcal{DL}$+log, and many others [15]. Hybrid knowledge representation systems that integrate OWL+SWRL and Prolog have also been proposed [16, 17]. We chose to address OWL limitations with SWRL and

Distribution A: Approved for public release; distribution is unlimited.

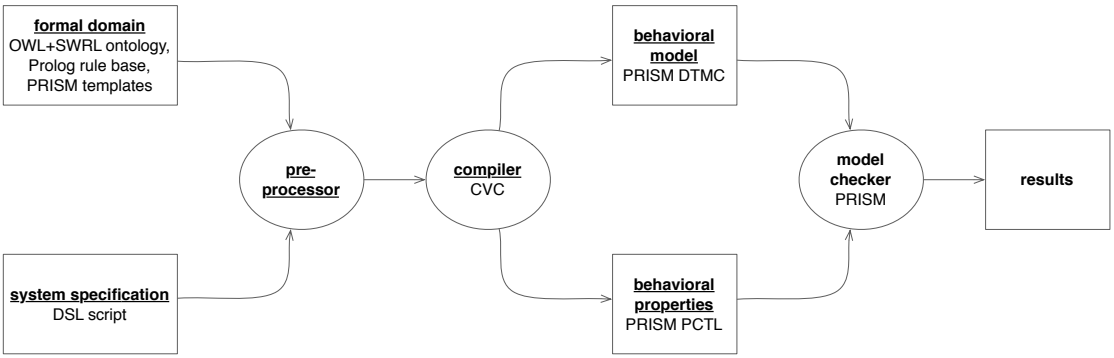

Figure 1: A high-level, domain-agnostic schematic of our method and prototype. Rectangular and oval shapes represent data and processes, respectively; bold and normal text distinguishes, respectively, method from prototype; and underlined text represents our research contributions with respect to the UAV domain.

Prolog; the former is an OWL extension approved by the W3C, while the latter is one of the most prominent logic-based knowledge representation languages.

Probabilistic model checking is supported by various software tools including ProbVerus and FMurφ, which analyze DTMC models; and ETMCC, MRMC (the successor of ETMCC), LiQuor and Rapture, which analyze *Markov decision process* (MDP) models [1]. But PRISM is, in our opinion, preferable because it supports both model types, thereby extending the potential of our method and prototype. PRISM also supports PCTL, a formalism that can express a large class of properties in an elegant manner.

3.1 Method Overview

Figure 1 illustrates a high-level, domain-agnostic schematic of our method and prototype. Domain experts, which are the method's primary stakeholders, use OWL to define domain concepts and their relationships; SWRL and Prolog to define rules; and PRISM's modeling and property specification languages to define, respectively, DTMC and PCTL templates. Model builders—also primary stakeholders—use a high-level DSL to encode system specifications that can be analyzed with model checking. We note that domain knowledge is formalized once and subsequently reused to support the verification of multiple system specifications.

3.2 An Example Mission

With our prototype, model builders use a domain-specific YAML dialect to encode mission plans comprising UAV *assets* and the *action workflows* assigned to those assets. The YAML code in Listing 1 specifies Mission A, an example mission that is representative of the 58 mission plans developed for this project.

Listing 1: YAML code for Mission A

Distribution A: Approved for public release; distribution is unlimited.

```
Action:                                                                  1
  TraversePathSegmentAction:                                             2
    - id: TPSA1                                                          3
      duration: 60                                                       4
      coordinates: [-118.27017, 34.04572,                                5
        -118.27279, 34.04284]                                            6
    - id: TPSA2                                                          7
      duration: 60                                                       8
      coordinates: [-118.2739, 34.03928]                                 9
      preconditions: [TPSA1, TPSA3]                                      10
    - id: TPSA3                                                          11
      duration: 60                                                       12
      coordinates: [-118.26482, 34.03332,                                13
        -118.27383, 34.03824]                                            14
    - id: TPSA4                                                          15
      duration: 60                                                       16
      coordinates: [-118.28204, 34.0376]                                 17
      preconditions: [TPSA3]                                             18
  PhotoSurveillanceAction:                                               19
    - id: PSA5                                                           20
      duration: 50                                                       21
      preconditions: [TPSA3]                                             22
Asset:                                                                   23
  Hummingbird:                                                           24
    - id: H1                                                             25
      actions: [TPSA1, TPSA2]                                            26
    - id: H2                                                             27
      actions: [TPSA3, TPSA4, PSA5]                                      28
```

Mission A comprises two *Hummingbird* assets (lines 24–28 in Listing 1); a single *photo surveillance action* (lines 19–22), which is a type of *sensor action*; and four *path segment traversal actions* (lines 2–18), which are *kinetic actions*. A path segment traversal action instructs the executing UAV to traverse a path between two waypoints. For each such action, the latitudes and longitudes of the delineating waypoints are stored in an array and indexed in succession; in other words, the latitude of waypoint one is followed by the longitude of waypoint one, which is in turn followed by the latitude of waypoint two, etc. We note that the end coordinates of an action a constitute the start coordinates of an action b if a precedes b, and both a and b are assigned to the same asset; for example, the end coordinates of action `TPSA1` (line 6) constitute the implied start coordinates of action `TPSA2`, which succeeds `TPSA1` in the sequence of kinetic actions assigned to asset `H1`.

3.3 From Specification to Verification

For any given mission specification, a *cascading verification compiler* (CVC) synthesizes both the DTMC and PCTL artifacts corresponding to that specification. Artifacts are synthesized as

Distribution A: Approved for public release; distribution is unlimited.

follows:

1. Mission specifications encoded in YAML are transformed by the CVC into ABox assertions. During this preprocessing phase, the CVC uses geographic coordinates from mission specifications, and data (pertaining to operational environments) from external sources, to perform geodetic calculations.[1] The equations that support these calculations are hard-coded in the CVC; for example, the compiler comprises geodesic equations that establish the occurrence, and calculate the duration, of threat area incursions committed by UAVs. Geographic information resulting from preprocessing is integrated with the generated ABox.

2. Pellet—a sound and complete OWL reasoner [18]—verifies the consistency of the generated ABox with respect to the TBox defined by domain experts. In doing so, the reasoner ensures that mission constructs encoded in YAML are consistent with OWL+SWRL axioms. Inconsistencies signify an invalid mission specification, which causes the compilation process to terminate with an error. If consistency is deduced, then the reasoner proceeds to generate inferences from explicitly encoded domain knowledge; for example, if geodetic calculations establish the occurrence of a threat area incursion, then the asset committing that incursion is inferred to be a *threatened asset*.

3. Inferred ontological knowledge is transformed by the CVC into Prolog facts. The compiler for SWI-Prolog—an open source Prolog implementation [19]—inputs the generated fact-base and the Prolog rule-base defined by domain experts, and proceeds to generate inferences; for example, the last kinetic action in an action workflow is inferred to be a *default terminal action*. The CVC uses Prolog inferences, in conjunction with explicit and inferred ontological knowledge, to synthesize DTMC and PCTL artifacts from predefined templates.

PRISM inputs the synthesized artifacts, verifies the system model against its desired behavioral properties, and returns logical and probabilistic results from the verification. If the results are deemed acceptable by the model builder(s), then the mission can be scheduled for real-world execution (via some process that is outside the scope of our method).

4 Results and Discussion

We assert that by enhancing the abstraction level of model and property specifications, cascading verification also enhances the effectiveness of probabilistic model checking. This assertion is validated with a prototype implementation of cascading verification for the UAV domain. The prototype benefits mission developers by simplifying the verification of UAV mission plans, and by augmenting PRISM's verification capabilities. Ultimately, our prototype benefits mission developers by improving the correctness of UAV mission specifications.

[1] Geodetics is a branch of applied mathematics that deals with the size and shape of the Earth.

Distribution A: Approved for public release; distribution is unlimited.

4.1 Abstraction

Because it was unfeasible to involve practitioners in the evaluation of our prototype's utility, we opted instead for a metrics-based analysis of 58 mission plans. These plans were based on real-world mission scenarios developed independently by DARPA and DRDC [11, 12]. We evaluated our approach by comparing the lines of code (LOC) and numbers of lexical tokens required to specify missions in YAML, against the LOC and tokens in the combined DTMC and PCTL code synthesized by the CVC. On average, our prototype synthesizes PRISM code that is 3.127 and 4.490 times greater than the size of the YAML input with regard to LOC and tokens, respectively. (The standard deviations were 52.4% and 95.4%, respectively.) These results demonstrate a non-trivial reduction in the effort required to produce mission models and properties.

We observe that tactical missions generate more LOC and tokens than *standalone mission plans*, which are mission plans not associated with the more specialized tactical subdomain. Specifically, tactical mission plans generate PRISM code that is on average 3.933 and 5.992 times greater than the size of the YAML input with regard to LOC and tokens, respectively. (The standard deviations were 24.0% and 59.2%, respectively.) Because the effort required to synthesize PRISM code is proportional to the effort required to synthesize the LOC and tokens that constitute the code, tactical mission plans result in added value for mission developers. This observation suggests that, with respect to standalone and tactical missions, the utility of our prototype is proportional to the threat level associated with any given mission plan. More broadly, increased LOC and token output suggests that the utility of cascading verification may be proportional to the amount of automated reasoning required to synthesize pertinent artifacts, a conclusion that justifies our motivation to augment model checking with formalized domain knowledge.

4.2 Effectiveness

Because it cannot account for the intricate syntax of the PRISM language, a LOC- and token-based analysis offers limited insight into the inherent complexity of model and property specifications. We investigated complexity further by considering *behavioral modeling errors* specific to the PRISM language that can be eliminated by the automated synthesis of PRISM artifacts (at least with respect to the segment of the mission space that we have investigated thus far) [10]. These errors are significant—perhaps more so than the errors uncovered during the model checking process—because they can mislead mission developers by causing PRISM to verify erroneous mission plans.

We also identified 28 *mission specification errors*, across six error classes, that impact the correctness of UAV missions and are beyond the scope of PRISM's verification capabilities. These errors are detected by either Pellet or the SWI-Prolog compiler during the synthesis process.

Mission correctness can clearly be compromised by mission specification and behavioral

Distribution A: Approved for public release; distribution is unlimited.

modeling errors, which occur during the design/implementation and verification phases, respectively, of the mission development process. Our prototype augments PRISM's effectiveness by preventing both of these error types.

4.3 Probabilistic Verification

We also considered PRISM's ability to meaningfully verify UAV mission plans (or rather, the utility of the DTMC and PCTL artifacts synthesized by our prototype). For this part of the evaluation, eighteen of the 58 mission plans described above were seeded with errors—including deadlock and non-reachable states—that violated desirable behavioral properties. One mission plan failed (i.e., contained errors that resulted in a 0.0 probability of success) because of an unacceptably low RAF value; nine mission plans failed because kinetic or sensor action workflow durations exceeded the endurances of the assets to which those workflows were assigned; and eight mission plans contained action workflow errors that resulted in deadlock. These errors were successfully identified by our prototype. While the correctness of some mission plans was absolute (with a 0.0 or 1.0 probability of success) several mission plans, including plans comprising threat area incursions, were associated with variable probabilities of success.

4.4 Discussion

By automating the synthesis of PRISM artifacts, and by providing multiple stages of reasoning and analysis, our prototype enhances the abstraction level of model and property specifications, and the effectiveness of probabilistic model checking, respectively. This cascading approach to verification improves mission correctness to a degree that is evidently unattainable by the individual components that constitute the prototype.

We note that the evaluation presented in this section is preliminary. Further work is required to determine the utility of our prototype in the context of a more sophisticated mission specification language and domain model.

5 Outputs from the Project

The research outlined in this report was the outcome of collaborative research. David Rosenblum and Fokion Zervoudakis have visited the University of Nebraska-Lincoln during the course of the project. The outcome of our research was a refereed conference paper, which was accepted for publication by the 9th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2013). In addition, Fokion Zervoudakis, the Ph.D. student who trained under the project, is expected to submit his thesis to the Department of Computer Science at University College London (UCL) in September-October 2013.

Distribution A: Approved for public release; distribution is unlimited.

6 Conclusions

We have identified several promising directions for future work. Composite CVC inferences are currently unidirectional, with Prolog facts derived from knowledge encoded in OWL+SWRL. While conceptually and practically appealing, this *pipeline* architecture constrains the reasoning process from refining Prolog inferences with ontological knowledge, and increases the potential for knowledge duplication. We aim to address these limitations by developing a knowledge representation framework that can support more flexible, iterative reasoning.

A second issue pertains to the artifacts that constitute the CVC knowledge base, including the ontology, the Prolog rule-base, and the DTMC and PCTL templates. These artifacts should be extensible to reflect changes in domain knowledge. Extensions should in turn be verifiable to ensure that domain knowledge remains consistent across the entire knowledge base. This requirement provides impetus for the development of a mechanism that will automate the consistency management process.

We also intend to further the evaluation of our method and prototype by enhancing the sophistication of the mission specification language and domain model presented in this report. We expect a more robust evaluation process to facilitate the abstraction, and formal specification, of the connections that link different technologies in the context of our method. If formalized, these connections would support the consistency management process described above.

6.1 Network-Centric Operations

Network-Centric Operations (NCO) is a military doctrine that aims to improve the efficiency and effectiveness of U.S. combat operations by leveraging information technology [20]. NCO is underpinned in part by contemporary socio-technological advancements, and enabled by a high-performance information grid; access to appropriate information sources; weapons reach and maneuver with precision and speed of response; *command-and-control* (C2) processes that support high-speed assignment of resources to need; and integrated sensor grids that are closely coupled to C2 processes and shooters [21]. When combined, these elements support speed of command, the process by which a superior information position is turned into a competitive advantage. Speed of command can be substantially enhanced when command-and-control processes are automated. Enhanced speed of command accelerates the *observe, orient, decide and act* (OODA) loop, which denies the enemy operational pause. Regaining this time amplifies the effects associated with speed of command, resulting in an accelerated rate of change that leads to enemy lock-out.

By automating the organization and utilization of military knowledge, Semantic Web technologies could support the verification and deployment of mission plans comprising asset configurations derived from real-time operational data, including asset location, fuel and weapon statuses. The near real-time coupling of mission verification and deployment has the potential to support a near real-time OODA loop. But the command-and-control process resulting from this

Distribution A: Approved for public release; distribution is unlimited.

coupling will inevitably be susceptible to network and processing speed latencies. Addressing the impact of latency on mission dependability in the context of NCO constitutes an interesting research direction.

References

[1] C. Baier and J.-P. Katoen, *Principles of Model Checking*. The MIT Press, 2008.

[2] Robby, M. B. Dwyer, and J. Hatcliff, "Bogor: An Extensible and Highly-Modular Software Model Checking Framework," in *Proceedings of the 9th European Software Engineering Conference Held Jointly With the 11th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ESEC/FSE-11, pp. 267–276, ACM, 2003.

[3] S. Narayanan and S. A. McIlraith, "Simulation, Verification and Automated Composition of Web Services," in *Proceedings of the 11th International Conference on World Wide Web*, WWW '02, pp. 77–88, ACM, 2002.

[4] I. D. Pietro, F. Pagliarecci, and L. Spalazzi, "Model Checking Semantically Annotated Services," *IEEE Transactions on Software Engineering*, vol. 38, pp. 592–608, May 2012.

[5] L. Boaro, E. Glorio, F. Pagliarecci, and L. Spalazzi, "Semantic Model Checking Security Requirements for Web Services," in *Proceedings of the 2010 International Conference on High Performance Computing and Simulation*, HPCS '10, pp. 283–290, IEEE, 2010.

[6] G. Oghabi, J. Bentahar, and A. Benharref, "On the Verification of Behavioral and Probabilistic Web Services Using Transformation," in *Proceedings of the 2011 IEEE International Conference on Web Services*, ICWS '11, pp. 548–555, IEEE Computer Society, 2011.

[7] A. Ankolekar, M. Paolucci, and K. Sycara, "Towards a Formal Verification of OWL-S Process Models," in *Proceedings of the 4th International Conference on the Semantic Web*, ISWC '05, pp. 37–51, Springer-Verlag, 2005.

[8] R. Liu, C. Hu, and C. Zhao, "Model Checking for Web Service Flow Based on Annotated OWL-S," in *Proceedings of the 2008 Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, SNPD '08, pp. 741–746, IEEE Computer Society, 2008.

[9] A. Lomuscio and M. Solanki, "Mapping OWL-S Processes to Multi Agent Systems: A Verification Oriented Approach," in *Proceedings of the 2009 International Conference on Advanced Information Networking and Applications Workshops*, WAINA '09, pp. 488–493, IEEE Computer Society, 2009.

[10] F. Zervoudakis, D. Rosenblum, S. Elbaum, and A. Finkelstein, "Cascading Verification: An Integrated Method for Domain-Specific Model Checking," in *Proceedings of the 9th*

Distribution A: Approved for public release; distribution is unlimited.

Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE '13, ACM, 2013.

[11] DARPA, "UAVForge." [online] Available at: `http://www.uavforge.net/`.

[12] G. Youngson, K. Baker, D. Kelleher, and S. Williams, "Project Support Services for the Operational Mission and Scenario Analysis for Multiple UAVs/UCAVs Control From Airborne Platform," March 2004.

[13] C. C. Evans, "The Official YAML Web Site." [online] Available at: `http://yaml.org/`.

[14] I. Horrocks and P. F. Patel-Schneider, "Knowledge Representation and Reasoning on the Semantic Web: OWL," in *Handbook of Semantic Web Technologies* (J. Domingue, D. Fensel, and J. A. Hendler, eds.), ch. 9, pp. 365–398, Springer, 2011.

[15] B. Motik, I. Horrocks, R. Rosati, and U. Sattler, "Can OWL and Logic Programming Live Together Happily Ever After?," in *Proceedings of the 5th International Conference on the Semantic Web*, ISWC '06, pp. 501–514, Springer-Verlag, 2006.

[16] T. Matzner and P. Hitzler, "Any-World Access to OWL From Prolog," in *Proceedings of the 30th Annual German Conference on Advances in Artificial Intelligence*, KI '07, pp. 84–98, Springer-Verlag, 2007.

[17] K. Samuel, L. Obrst, S. Stoutenberg, K. Fox, P. Franklin, A. Johnson, K. Laskey, D. Nichols, S. Lopez, and J. Peterson, "Translating OWL and Semantic Web Rules Into Prolog: Moving Toward Description Logic Programs," *Theory and Practice of Logic Programming*, vol. 8, pp. 301–322, May 2008.

[18] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: A practical OWL-DL Reasoner," *Journal of Web Semantics*, vol. 5, pp. 51–53, June 2007.

[19] "SWI-Prolog's home." [online] Available at: `http://www.swi-prolog.org/`.

[20] C. Wilson, "Network Centric Operations: Background and Oversight Issues for Congress," March 2007.

[21] A. K. Cebrowski and J. J. Garstka, "Network-Centric Warfare: Its Origin and Future," *U.S. Naval Institute Proceedings*, vol. 124, no. 1, pp. 28–35, 1998.

Distribution A: Approved for public release; distribution is unlimited.